# PLP - Segundo Parcial - 2$^{do}$ cuatrimestre de 2024

| #Orden | Nro. Libreta | Apellido y Nombre | Ej1 | Ej2 | Ej3 | Nota Final |
|---|---|---|---|---|---|---|
| | | | MB | MB | MB | |

Este examen se aprueba obteniendo al menos dos ejercicios bien (B) y uno regular (R), y se promociona con al menos dos ejercicios muy bien (MB) y uno bien (B). Es posible obtener una aprobación condicional con un ejercicio muy bien (MB), uno bien (B) y uno insuficiente (I), pero habiendo entregado algo que contribuya a la solución del ejercicio. Las notas para cada ejercicio son: -, I, R, B, MB, E. Entregar cada ejercicio en hojas separadas. Poner nombre, apellido y número de orden en todas las hojas, y numerarlas. Se puede utilizar todo lo definido en las prácticas y todo lo que se dio en clase, colocando referencias claras. El orden de los ejercicios es arbitrario. Recomendamos leer el parcial completo antes de empezar a resolverlo.

## Ejercicio 1 - Programación Lógica

Implementar los predicados respetando en cada caso la instanciación pedida. Los generadores deben cubrir todas las instancias válidas de aquello que generan sin repetir dos veces la misma. Se deben indicar los patrones de instanciación de todos los predicados auxiliares. No usar cut (`!`) ni predicados de alto orden como `setof`, con la única excepción de `not`.

a) Definir el predicado `subsecuenciaCreciente(+L,-S)` que es verdadero cuando `S` es una subsecuencia estrictamente creciente de elementos de `L`. Notar que la secuencia respeta el orden de aparición en `L`. Por ejemplo:

```
?- subsecuenciaCreciente([4,8,1,9],S).
 S = [4, 8, 9] ;
 S = [4, 8] ;
 S = [4, 9] ;
 S = [4] ;
 S = [8, 9] ;
 S = [8] ;
 S = [1, 9] ;
 S = [1] ;
 S = [9] ;
 S = [].
```

b) Definir el predicado `subsecuenciaCrecienteMasLarga(+L,-S)` que es verdadero cuando `S` es la subsecuencia estrictamente creciente de mayor longitud de `L`. Puede haber más de un resultado. Por ejemplo:

```
?- subsecuenciaCrecienteMasLarga([5,6,7,2,8,1,2,3,4,5,7],S).
 S = [1,2,3,4,5,7] ;
 false;

?- subsecuenciaCrecienteMasLarga([5,6,7,2,8,0,2,3,4],S).
 S = [5,6,7,8] ;
 S = [0,2,3,4] ;
 false.
```

c) Definir el predicado `fibonacci(-X)` que instancia en `X` los números pertenecientes a la secuencia de Fibonacci. Por ejemplo:

```
?- fibonacci(X).
 X = 1;
 X = 1;
 X = 2;
 X = 3;
 X = 5;
 ...
```

d) ¿Es reversible el predicado anterior? Justificar.

## Ejercicio 2 - Resolución

a) Representar en forma clausal las siguientes fórmulas de lógica de primer orden, que tratan acerca de términos cerrados del cálculo lambda:

i) $\forall T_1.\forall T_2.\forall M.\forall N.((\mathsf{Tipo}(M, T_1 \to T_2) \wedge \mathsf{Tipo}(N, T_1)) \implies \mathsf{Tipo}(\mathsf{app}(M, N), T_2))$
*Si el tipo de un término es $T_1 \to T_2$, y el tipo de otro término es $T_1$, entonces el tipo de la aplicación es $T_2$.*

ii) $\exists M.\mathsf{Tipo}(M, \alpha \to (\beta \to \gamma))$
*Existe un término de tipo $\alpha \to (\beta \to \gamma)$.*

iii) $\exists M.\mathsf{Tipo}(M, \alpha \to \beta)$
*Existe un término de tipo $\alpha \to \beta$.*

iv) $\exists M.\mathsf{Tipo}(M, \alpha)$

*Existe un término de tipo $\alpha$.*

*Ayuda: se puede pensar en el constructor de tipos $\rightarrow$ como una función binaria, por ejemplo el término $T_1 \rightarrow T_2$ se puede pensar como $\mathsf{flecha}(T_1, T_2)$. Además, $\alpha$, $\beta$ y $\gamma$ son constantes distintas entre sí.*

b) Utilizando resolución, determinar si la siguiente fórmula es consecuencia del conjunto anterior: $\exists M.\mathsf{Tipo}(M, \gamma)$ *(Existe un término de tipo $\gamma$).*

Indicar la sustitución utilizada en cada paso. Es importante tener un plan (escrito o en la cabeza).

c) ¿Fue SLD la resolución utilizada en el punto anterior? Justificar.

## Ejercicio 3 - Inferencia y Deducción Natural

a) Consideremos el Cálculo Lambda tipado extendido con árboles ternarios:

$\tau ::= \cdots \mid \mathtt{AT}(\tau)$

$M ::= \cdots \mid \mathtt{TNil}_\tau \mid \mathtt{Tern}(M, M, M, M) \mid \mathsf{foldAT}\ M \triangleright\ \mathsf{TNil}\ \rightsquigarrow M;\ \mathsf{Tern}(x, ri, rm, rd) \rightsquigarrow M$

Se extiende también el algoritmo de inferencia para árboles de la siguiente manera:

$\mathbb{W}(\mathtt{TNil}) \stackrel{def}{=} \emptyset \vdash \mathtt{TNil}_X : \mathtt{AT}(X)$ con $X$ variable fresca

$\mathbb{W}(\mathtt{Tern}(U_1, U_2, U_3, U_4)) \stackrel{def}{=} S\Gamma_1 \cup S\Gamma_2 \cup S\Gamma_3 \cup S\Gamma_4 \vdash S(\mathtt{Tern}(M_1, M_2, M_3, M_4)) : S\sigma_4$

donde:

- $\mathbb{W}(U_1) = \Gamma_1 \vdash M_1 : \sigma_1$
- $\mathbb{W}(U_2) = \Gamma_2 \vdash M_2 : \sigma_2$
- $\mathbb{W}(U_3) = \Gamma_3 \vdash M_3 : \sigma_3$
- $\mathbb{W}(U_4) = \Gamma_4 \vdash M_4 : \sigma_4$
- $S = \text{mgu}\ (\{\sigma_2 \stackrel{?}{=} \mathtt{AT}(\sigma_1),\ \sigma_3 \stackrel{?}{=} \sigma_2,\ \sigma_4 \stackrel{?}{=} \sigma_2\} \cup$
  $\{\tau \stackrel{?}{=} \rho \mid x : \tau \in \Gamma_i\ \wedge\ x : \rho \in \Gamma_j\ \wedge\ i, j \in \{1, 2, 3, 4\}\})$

$\mathbb{W}(\mathsf{foldAT}\ U_1 \triangleright\ \mathsf{TNil}\ \rightsquigarrow U_2;\ \mathsf{Tern}(x, ri, rm, rd) \rightsquigarrow U_3) \stackrel{def}{=} S\Gamma_1 \cup S\Gamma_2 \cup S\Gamma_{3'} \vdash S(\mathsf{foldAT}\ M_1 \triangleright\ \mathsf{TNil}\ \rightsquigarrow M_2;\ \mathsf{Tern}(x, ri, rm, rd) \rightsquigarrow M_3) : S\sigma_2$

donde:

- $\mathbb{W}(U_1) = \Gamma_1 \vdash M_1 : \sigma_1$
- $\mathbb{W}(U_2) = \Gamma_2 \vdash M_2 : \sigma_2$
- $\mathbb{W}(U_3) = \Gamma_3 \vdash M_3 : \sigma_3$
- $\tau_x = \begin{cases} \alpha_1 \text{ si } x : \alpha_1 \in \Gamma_3, \\ \text{variable fresca si no} \end{cases}$
- $\tau_{ri} = \begin{cases} \alpha_2 \text{ si } ri : \alpha_2 \in \Gamma_3, \\ \text{variable fresca si no} \end{cases}$
- $\tau_{rm} = \begin{cases} \alpha_3 \text{ si } rm : \alpha_3 \in \Gamma_3, \\ \text{variable fresca si no} \end{cases}$
- $\tau_{rd} = \begin{cases} \alpha_4 \text{ si } rd : \alpha_4 \in \Gamma_3, \\ \text{variable fresca si no} \end{cases}$
- $\Gamma_{3'} = \Gamma_3 \ominus \{x, ri, rm, rd\}$
- $S = \text{mgu}\ (\{\sigma_1 \stackrel{?}{=} \mathtt{AT}(\tau_x),\ \sigma_2 \stackrel{?}{=} \sigma_3,\ \tau_{ri} \stackrel{?}{=} \sigma_2,\ \tau_{rm} \stackrel{?}{=} \sigma_2,\ \tau_{rd} \stackrel{?}{=} \sigma_2\} \cup$
  $\{\tau \stackrel{?}{=} \rho \mid x : \tau \in \Gamma_i\ \wedge\ x : \rho \in \Gamma_j\ \wedge\ i, j \in \{1, 2, 3'\}\})$

Además, se considera extendido el algoritmo para la suma de naturales:

$\mathbb{W}(U_1 + U_2) \stackrel{def}{=} S\Gamma_1 \cup S\Gamma_2 \vdash S(M_1 + M_2) : Nat$

donde:

- $\mathbb{W}(U_1) = \Gamma_1 \vdash M_1 : \sigma_1$
- $\mathbb{W}(U_2) = \Gamma_2 \vdash M_2 : \sigma_2$
- $S = \text{mgu}\ (\{\sigma_1 \stackrel{?}{=} Nat,\ \sigma_2 \stackrel{?}{=} Nat\} \cup \{\tau \stackrel{?}{=} \rho \mid x : \tau \in \Gamma_1\ \wedge\ x : \rho \in \Gamma_2\})$

Inferir el tipo de la siguiente expresión, o demostrar que no es tipable:

$$\mathsf{foldAT}\ \mathtt{Tern}(\underline{1}, \mathtt{TNil}, x, x) \triangleright\ \mathsf{TNil}\ \rightsquigarrow rd;\ \mathsf{Tern}(x, ri, rm, rd) \rightsquigarrow (ri + rm) + rd$$

b) Demostrar el siguiente teorema usando deducción natural, sin utilizar principios clásicos:

$$(\exists X.P(X)) \Rightarrow (\forall Y.(P(Y) \Rightarrow Q(Y))) \Rightarrow \exists Z.Q(Z)$$

[redacted] [redacted] [redacted] 1

1) a). Podríamos en principio obtener todas las subsecuencias de L, siguiendo la siguiente estrategia: por cada elemento de la lista, siguiéndolo en orden consideramos ponerlo o no ponerlo en la subsecuencia. Luego de esto, nos quedamos con aquellas que son crecientes

- subsecuenciaCreciente(L, S) :- subsecuencia(L,S), esCreciente(S).

. subsecuencia([],[]).
subsecuencia([X|XS],[X|ZS]) :- subsecuencia(XS,ZS).
subsecuencia([_|XS], ZS) :- subsecuencia(XS, ZS).

(subsecuencia(+L, -S))

. esCreciente([]).
esCreciente([X]).
esCreciente([X|Y|XS]) :- X< Y, esCreciente([Y|XS]).

(esCreciente(+L))

a esta altura está instanciado, así que es válida la comparación

b). Podríamos pedir que S sea subsecuencia creciente y que además no exista otra cuyo longitud sea mayor (y sea creciente)

- subsecuenciaCrecienteMasLarga(L, S) :-
subsecuenciaCreciente(L, S), length(S, L1)
not((subsecuenciaCreciente(L, ZS), length(ZS, L2), L2 > L1))

c). Podemos utilizar el hecho de que un número en la secuencia de Fibonacci es la suma de los dos anteriores. Podemos saber el M-ésimo número de Fibonacci.
- Fibonacci(N) :- desde(0, M), FibonacciM(M, N) (FibonacciM(M+, N-))

. FibonacciM(0,1).
FibonacciM(1,1).
FibonacciM(M, N) :- M \= 0, M \= 1, M1 is M-1, M2 is M-2, FibonacciM(M1, S1),
FibonacciM(M2, S2), N is S1+S2.

M instanciado

No No
d) Sí, creo que es reversible. Si instanciamos el N, entonces va a buscar un M tal que ese N sea el M-ésimo número de la secuencia de Fibonacci. El problema es que si el número no es de Fibonacci, entonces el M se nos puede ir al infinito y el programa se colgaría.

(*) desde(X,X).
desde(X,Y) :- N is X+1, desde(N,Y).

(desde(+X, -Y))

No es reversible ya que se va a ejecutar infinitamente la tercer regla

NOTA

[redacted] [redacted] [redacted]

HOJA N° 3

FECHA

2) a) $\forall T_1. \forall T_2. \forall M. \forall N. ((\text{Tipo}(M, T_1 \to T_2) \land \text{Tipo}(N, T_1)) \Rightarrow \text{Tipo}(\text{app}(M,N), T_2))$

$= \forall T_1. \forall T_2. \forall M. \forall N. (\neg(\text{Tipo}(M, T_1 \to T_2) \land \text{Tipo}(N, T_1)) \lor \text{Tipo}(\text{app}(M,N), T_2))$

$= \forall T_1. \forall T_2. \forall M. \forall N. ((\neg \text{Tipo}(M, T_1 \to T_2) \lor \neg \text{Tipo}(N, T_1)) \lor \text{Tipo}(\text{app}(M,N), T_2))$

1. ① $\{\{\neg \text{Tipo}(M, T_1 \to T_2), \neg \text{Tipo}(N, T_1), \text{Tipo}(\text{app}(M,N), T_2)\}\}$

Como comentario, ☆ cuando obtenes una cláusula resolvente, las variables de dicha cláusula son variables frescas, de forma que ya no son las $T_i, N, M$ de antes, las podes distinguir como $N', N'', \dots$, o $N_1, N_2, \dots$

$\exists M. \text{Tipo}(M, \alpha \to (\beta \to \gamma))$

$= \text{Tipo}(c, \alpha \to (\beta \to \gamma))$

2. ② $\{\{\text{Tipo}(c, \alpha \to (\beta \to \gamma))\}\}$

$\exists M. \text{Tipo}(M, \alpha \to \beta)$

$= \text{Tipo}(d, \alpha \to \beta)$

3. ③ $\{\{\text{Tipo}(d, \alpha \to \beta)\}\}$

$\exists M. \text{Tipo}(M, \alpha)$

$= \text{Tipo}(e, \alpha)$

4. ④ $\{\{\text{Tipo}(e, \alpha)\}\}$

b) • Q.V.Q. $\exists M. \text{Tipo}(M, \gamma)$. Para probar la afirmación, la negamos y la agregamos junto con las otras cláusulas.

$\neg(\exists M. \text{Tipo}(M, \gamma))$

$= \forall M. \neg(\text{Tipo}(M, \gamma))$

⑤ $\{\{\neg(\text{Tipo}(M_5, \gamma))\}\}$

• El plan es el siguiente: sabemos que existe un término de tipo $\alpha$. También sabemos que existe un término de tipo $\alpha \to \beta$. Sabemos entonces que el tipo de la aplicación es $\beta$. Como sabíamos que $\alpha \to (\beta \to \gamma)$ y sabemos que hay uno con tipo $\beta$ (la aplicación), entonces va a existir uno con tipo $\gamma$.

C = {①,②,②,④,⑤}

1 y 3: $S = \{M := d, T_1 := \alpha, T_2 := \beta\}$ ⑥ $\{\neg \text{Tipo}(N^{☆}, \alpha), \text{Tipo}(\text{app}(d, N^{☆}), \beta)\}$

4 y 6: $S = \{N := e\}$ ⑦ $\{\text{Tipo}(\text{app}(d,e), \beta)\}$

1 y 4: $S = \{N := e, T_1 := \alpha\}$ ⑧ $\{\neg \text{Tipo}(M^{☆}, \alpha \to T_2^{☆}), \text{Tipo}(\text{app}(M^{☆}, e), T_2^{☆})\}$

8 y 2: $S = \{M := c, T_2 := (\beta \to \gamma)\}$ ⑨ $\{\text{Tipo}(\text{app}(c,e), (\beta \to \gamma))\}$

9 y 1: $S = \{M := \text{app}(c,e), T_1 := \beta, T_2 := \gamma\}$ ⑩ $\{\neg \text{Tipo}(N^{☆}, \beta), \text{Tipo}(\text{app}(\text{app}(c,e), N^{☆}), \gamma)\}$

10 y 7: $S = \{N := \text{app}(d,e)\}$ ⑪ $\{\text{Tipo}(\text{app}(\text{app}(c,e), \text{app}(d,e)), \gamma)\}$

11 y 5: $S = \{M_5 := \text{app}(\text{app}(c,e), \text{app}(d,e))\}$ ⑫ $\{\}$

• Como llegamos a la cláusula vacía, entonces la afirmación: "Existe un término de tipo $\gamma$" era válida, pues encontramos ese término. Notar que esto fue porque llegamos a que aplicaciones de aplicaciones de CONSTANTES tienen tipo $\gamma$.

No era válida por sí sola, sino que se dedujo de la base de conocimientos obtenida

c) • Para que sea una resolución SLD, se tienen que cumplir 4 propiedades: Empezar por una cláusula objetivo, que sean todas cláusulas de Horn, que la resolución sea lineal y que sea binaria. En particular, no fue lineal, entonces no fue una resolución SLD.

NOTA

[redacted] [redacted] HOJA Nº 2

FECHA

3) • Inferir el tipo de:

12) FoldAT Tern(1, TNil, x, x) ▷ TNil → rd ; Tern(x, ri, rm, rd) → (ri + rm) + rd

9) Tern(1, TNil, x, x) 10) rd 11) (ri + rm) + rd

1) 1 2) TNil 3) x 4) x 7) (ri + rm) 8) rd

0 — succ(0) 5) ri 6) rm

Faltó un paso más, la regla de esta manera no existe

1) $W(1) \rightarrow \phi \vdash 1 : Nat$
2) $W(TNil) \rightarrow \phi \vdash TNil_{t_1} : AT(t_1)$
3) $W(x) \rightarrow x : t_2 \vdash x : t_2$
4) $W(x) \rightarrow x : t_3 \vdash x : t_3$
5) $W(ri) \rightarrow ri : t_4 \vdash ri : t_4$
6) $W(rm) \rightarrow rm : t_5 \vdash rm : t_5$
7) $W(ri + rm) \rightarrow S = mgu\{t_4 \doteq Nat, t_5 \doteq Nat\}$
$= \{t_5 \doteq Nat\}$ elim $\{t_4 := Nat\}$
$= \{\}$ elim $\{t_5 := Nat\}$
$S = \{t_4 := Nat, t_5 := Nat\}$
$\rightarrow ri : Nat, rm : Nat \vdash ri + rm : Nat$
8) $W(rd) \rightarrow rd : t_6 \vdash rd : t_6$
9) $W(Tern(1, TNil, x, x)) \rightarrow S = mgu\{AT(t_1) \doteq AT(Nat), t_2 \doteq AT(t_1), t_3 \doteq AT(t_1), t_2 \doteq t_3\}$
$= mgu\{t_1 \doteq Nat, t_2 \doteq AT(t_1), t_3 \doteq AT(t_1), t_2 \doteq t_3\}$ decompose
$= mgu\{t_2 \doteq AT(Nat), t_3 \doteq AT(Nat), t_2 \doteq t_3\}$ elim $\{t_1 := Nat\}$
$= mgu\{t_3 \doteq AT(Nat), AT(Nat) \doteq t_3\}$ elim $\{t_2 := AT(Nat)\}$
$= mgu\{AT(Nat) \doteq AT(Nat)\}$ elim $\{t_3 := AT(Nat)\}$
$= mgu\{\}$ delete
$S = \{t_1 := Nat, t_2 := AT(Nat), t_3 := AT(Nat)\}$
$\rightarrow x : AT(Nat) \vdash Tern(1, TNil_{Nat}, x, x) : AT(Nat)$

en realidad tendría que haber usado que $t_2 = t_3$. Entonces me quedaba uno solo de los dos. OK

10) $W(rd) \rightarrow rd : t_7 \vdash rd : t_7$
11) $W((ri + rm) + rd) \rightarrow S = mgu\{Nat \doteq Nat, t_6 \doteq Nat\}$
$= mgu\{t_6 \doteq Nat\}$ delete
$= mgu\{\}$ elim $\{t_6 := Nat\}$
$\rightarrow ri : Nat, rm : Nat, rd : Nat \vdash (ri + rm) + rd : Nat$
12) $W(FoldAT\ Tern(1, TNil, x, x) \triangleright TNil \rightarrow rd ; Tern(x, ri, rm, rd) \rightarrow (ri + rm) + rd) \rightarrow$
$S = mgu\{AT(Nat) \doteq AT(t_8), t_7 \doteq Nat, Nat \doteq t_7, Nat \doteq t_7, Nat \doteq t_7\}$
$= mgu\{Nat \doteq t_8, t_7 \doteq Nat, Nat \doteq t_7, Nat \doteq t_7, Nat \doteq t_7\}$ decompose
$= mgu\{t_8 \doteq Nat, t_7 \doteq Nat, Nat \doteq t_7, Nat \doteq t_7, Nat \doteq t_7\}$ swap
$= mgu\{t_7 \doteq Nat, Nat \doteq t_7, Nat \doteq t_7, Nat \doteq t_7\}$ elim $\{t_8 := Nat\}$
$= mgu\{Nat \doteq Nat, Nat \doteq Nat, Nat \doteq Nat\}$ elim $\{t_7 := Nat\}$
$= mgu\{\}$ elimx3
$S = \{t_7 := Nat, t_8 := Nat\}$

$\rightarrow x : AT(Nat), rd : Nat \vdash FoldAT\ Tern(1, TNil_{Nat}, x, x) \triangleright TNil \rightarrow rd ; Tern(x, ri, rm, rd) \rightarrow (ri + rm) + rd : Nat$

NOTA

b)

un poco desprolijo

$$
\dfrac{
\dfrac{}{\Gamma \vdash \exists x.\, P(x)}\,\text{Ax}
\qquad
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{\Gamma, P(x) \vdash \forall x.\, P(x) \Rightarrow Q(x)}\,\text{Ax}
}{\Gamma, P(x) \vdash P(\cancel{x}) \Rightarrow Q(\cancel{x})}\,\forall e
\qquad
\dfrac{}{\Gamma, P(x) \vdash P(x)}\,\text{Ax}
}{\Gamma, P(x) \vdash Q(\cancel{x})}\,\Rightarrow e
}{\Gamma, P(x) \vdash \exists z.\, Q(z)}\,\exists i
}{\Gamma = (\exists x.\, P(x)), (\forall y.\, (P(y) \Rightarrow Q(y))) \vdash \exists z.\, Q(z)}\,\exists e
$$

$$
\dfrac{\Gamma = (\exists x.\, P(x)), (\forall y.\, (P(y) \Rightarrow Q(y))) \vdash \exists z.\, Q(z)}{
\dfrac{(\exists x.\, P(x)) \vdash (\forall y.\, (P(y) \Rightarrow Q(y))) \Rightarrow \exists z.\, Q(z)}{(\exists x.\, P(x)) \Rightarrow ((\forall y.\, (P(y) \Rightarrow Q(y))) \Rightarrow \exists z.\, Q(z))}\,\Rightarrow i
}\,\Rightarrow i
$$

el un x